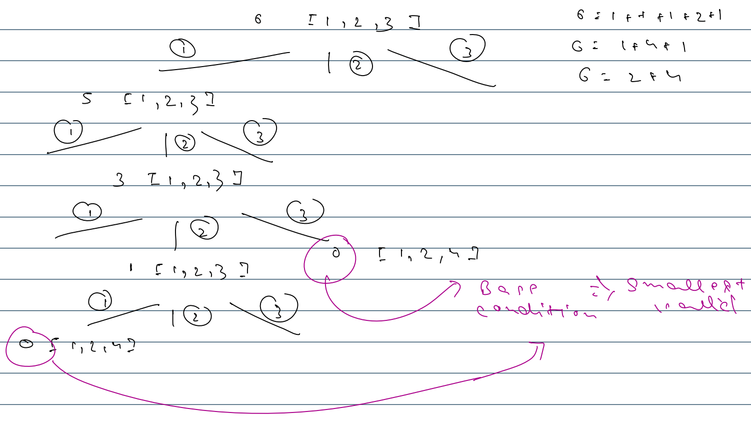
28 September 2025 10:07

Recursion and backtracking

coin change ⇒ [1, 3, 5] coins, amount 6 ↳ 1+1+1+3

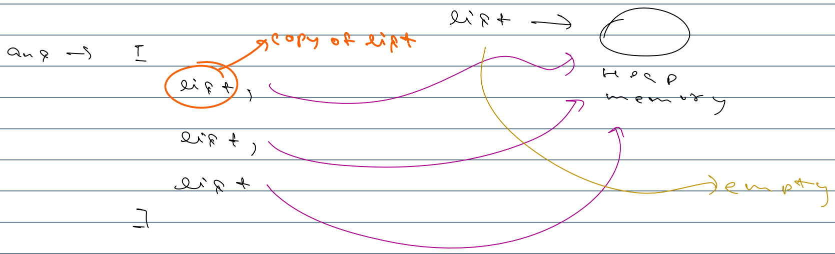
Recursion ⇒ Choices + Decision



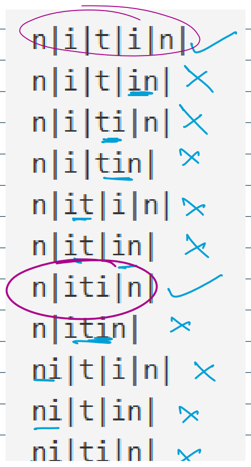
Combination Sum

o/p :- [1, 2, 3] → list of integers

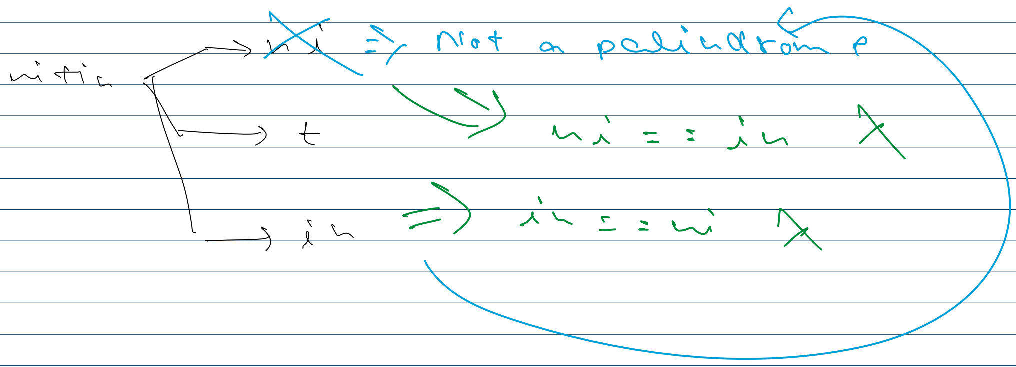
```
[
  [ 1, 2, 3 ],
  [ 1, 1 ],      → list of ~~list~~ → list of integers
  [ 1, 2 ]
]
```

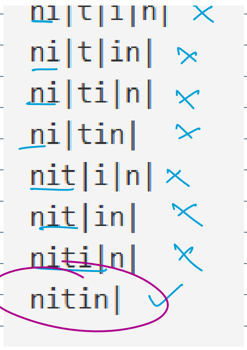


Palindrome Partitioning



n|i|t|i|n| ✓
n|i|t|in| ✗
n|i|ti|n| ✗
n|i|tin| ✗
n|it|i|n| ✗
n|it|in| ✗
n|iti|n| ✓
n|itin| ✗
ni|t|i|n| ✗
ni|t|in| ✗
ni|ti|n| ✗





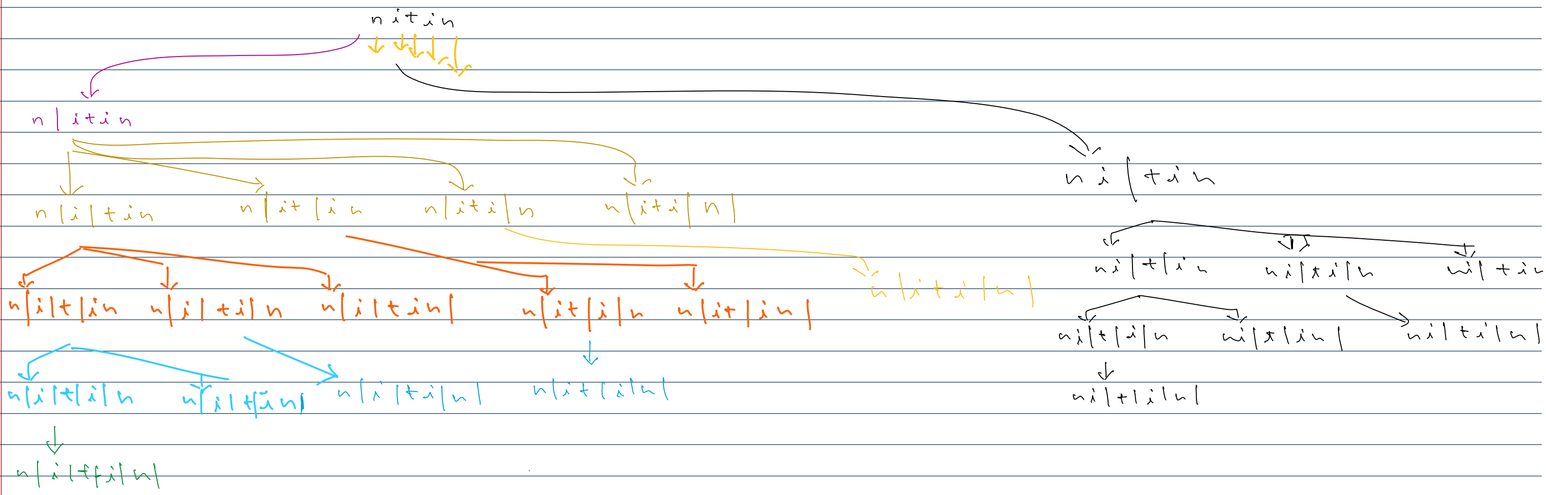
```
ni|t|i|n|   ✗
ni|t|in|    ✗
ni|ti|n|    ✗
ni|tin|     ✗
nit|i|n|    ✗
nit|in|     ✗
niti|n|     ✗
nitin|      ✓
```

itin ⇒ i   tin

it|in ⇒ it   in

it|in = iti   n

itin| = itin

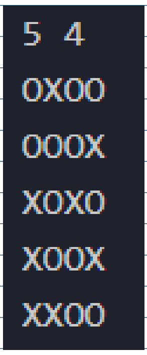


0 1 2 3 4
n i t i n
↓ ↓ ↓ ↓ ↓
① ② ③ ④ ⑤ ⇒ cnt

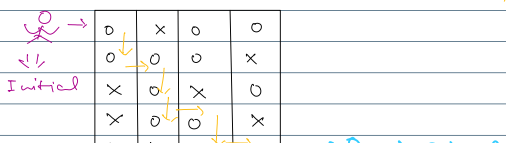
cnt = 1
cnt = 2
cnt = 3
cnt = 4
cnt = 5

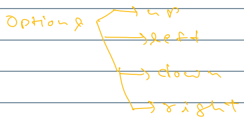
	prefix	rest
	n substring(0,1)	itin sub(1)
	ni substring(0,2)	tin sub(2)
	nit substring(0,3)	~~t~~in sub(3)
	niti sub(0,4)	n sub(4)
	nitin sub(0,5)	_ sub(5)

Rat chases its cheese

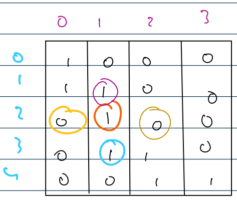


```
5 4
0X00
000X
X0X0
X00X
XX00
```



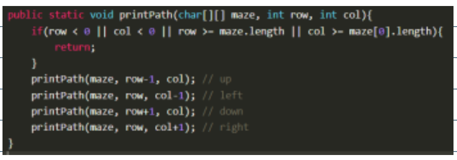


1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	1

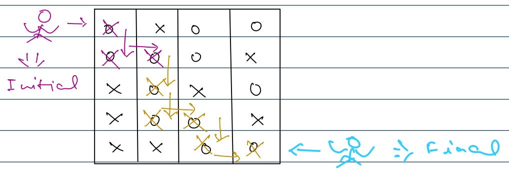


(2, 1)
(row, col) → right ⇒ (2, 2) ⇒ (row, col+1)
→ up ⇒ (1, 1) ⇒ (row-1, col)
→ down ⇒ (3, 1) ⇒ (row+1, col)
→ left ⇒ (2, 0) ⇒ (row, col-1)

Invalid options ⇒ row < 0
col < 0
row >= maze.length
col >= maze[0].length



```java
public static void printPath(char[][] maze, int row, int col){
    if(row < 0 || col < 0 || row >= maze.length || col >= maze[0].length){
        return;
    }
    printPath(maze, row-1, col); // up
    printPath(maze, row, col-1); // left
    printPath(maze, row+1, col); // down
    printPath(maze, row, col+1); // right
}
```



```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m = sc.nextInt();

        char[][] maze = new char[n][m];
        int[][] ans = new int[n][m];

        for(int i=0; i<n; i++){
            String s = sc.next();
            for(int j=0; j<m; j++){
                maze[i][j] = s.charAt(j);
            }
        }
        // displayChar(maze);
        // System.out.println("---------");

        printPath(maze, 0, 0, ans);

        if(valueFound == false){
            System.out.println("NO PATH FOUND");
        }
    }

    static boolean valueFound = false;

    public static void printPath(char[][] maze, int row, int col, int[][] ans){
        if(row==maze.length-1 && col==maze[0].length-1 && maze[row][col] != 'X'){
            ans[row][col] = 1;
            valueFound = true;
            displayInt(ans);
            return;
        }
        if(row < 0 || col < 0 || row >= maze.length || col >= maze[0].length || maze[row][col] == 'X'){
            return;
        }
        maze[row][col] = 'X';
        ans[row][col] = 1;
        int[] rows = {-1, 0, 1, 0};
        int[] cols = {0, -1, 0, 1};
        for(int i=0; i<4; i++){
            printPath(maze, row+rows[i], col+cols[i], ans);
        }
        // printPath(maze, row-1, col, ans); // up
        // printPath(maze, row, col-1, ans); // left
        // printPath(maze, row+1, col, ans); // down
        // printPath(maze, row, col+1, ans); // right
        maze[row][col] = 'O';
        ans[row][col] = 0;
    }


    public static void displayChar(char[][] arr){
        for(int i=0; i<arr.length; i++){
            for(int j=0; j<arr[0].length; j++){
```

```
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }

    public static void displayInt(int[][] arr){
        for(int i=0; i<arr.length; i++){
            for(int j=0; j<arr[0].length; j++){
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```